Semantic Web 0 (0) 1
IOS Press


# A pattern language for smart home applications

Marjan Alirezaie [a,*], Karl Hammar [b], Eva Blomqvist [b]
[a] *Center for Applied Autonomous Sensor Systems Örebro Universitet , Örebro, Sweden -* marjan.alirezaie@oru.se
[b] *SICS - East Swedish ICT -* karl@karlhammar.com, eva.blomqvist@liu.se

**Abstract.** In this article we outline the details of a SmartHome ontology proposed as a representational model to assist the development process of sensorized environments. The SmartHome ontology is described in terms of its modules representing different aspects including physical and conceptual aspects of a smart environment. We propose the use of the Ontology Design Pattern (ODP) paradigm in order to modularize our proposed solution, while at the same time avoiding strong dependencies between the modules in order to manage the representational complexity of the ontology. The ODP paradigm and related methodologies enable incremental construction of ontologies by first creating and then linking small modules. Most modules (patterns) of the SmartHome ontology are inspired by, and aligned with, the Semantic Sensor Network (SSN) ontology, however with slightly different interlinks to address a number of shortcomings of SSN and provide further precision to the definition of entities which would otherwise be modelled in several ways and consequently lead to representational uncertainty.The result is a network of 10 modules, which can be considered as ODPs, together forming a pattern language for the smart home domain. The ODPs have been submitted to the ODP portal and are available online at stable URIs.

Keywords: Smart Home Ontology, Ontology Design Pattern, Semantic Sensor Network

## 1. Introduction

Applications of sensorized environments and robots that provide domestic monitoring and cognitive assistance services are increasing. An example of such an application is healthcare monitoring and services, where patients are being monitored and cared for in their own home. In order to support the use of artificial intelligence techniques for automating the provision of these services, it is necessary to describe the capabilities of the various aspects of such environments. These descriptions include physical aspects (e.g., the structure of the environment, sensor network setting or entities in the environment), as well as conceptual aspects (e.g., events or daily activities of the users), and can be modelled in ontologies. According to [1], there is a general list of questions about the inhabitants of smart homes, which many of the suggested knowledge models in the literature aim to address. This list includes questions about the location of the inhabitant, the activity that the inhabitant carries out, the intention behind the activity, the time when the activity is detected, etc. Although the representational models (i.e., ontologies) target the same goal, they differ in terms of the level of generality as well as their reasoning efficiency.

Due to the dependencies between the aforementioned features, such an ontology can easily become large and complex; moreover, it may need to be updated over time, e.g., when sensors/robots with new kinds of features are added to (or removed from) the environment, or when the monitoring requirements of the environment change. Additionally, when we use ontologies in a system that requires near real-time reasoning and reactions by the system, the reasoning complexity is an essential parameter of the ontologies. For these reasons, we propose the use of a network of ontology modules, which could be considered

*

1570-0844/0-1900/$35.00 © 0 – IOS Press and the authors. All rights reserved

as a set of interlinked content Ontology Design Patterns (ODPs) [2] due to their generic applicability to the domain and our deliberate effort to minimize their ontological commitments, while maintaining functionality. According to the principles of ODP modelling, the ontological commitments should be minimized by first creating small modules, and then linking them together, instead of designing a large monolithic ontology from scratch as a comprehensive representation of the entire domain [3].

Furthermore, we view our set of ODPs as a pattern language for ontology-based smart home applications. The term pattern language was first coined by C. Alexander in the domain of built environments [4], but has since been adopted by many fields, including computer science. Generally the term refers to a set of patterns that are interlinked, i.e. have explicit relations to each other, and can be used together to solve design problems within a certain domain.

### *1.1. Overview on Sensor-Related Ontologies*

Sensing and in general sensor-related details of smart homes as one of their integral computational aspects has been widely studied [5]. Although the "reusability" has always been advertised as an essential features of ontologies, there is a large number of adhocly designed sensor-related ontologies in the literature, many of which are not even available online. This number increases when the idea behind the design of these ontologies is getting more specific, for example for activity recognition purposes. Many of the ontologies suggested for recognising daily activities ([6,7]) are barely relying on upper-level ontologies, which results in a lower chance of reusability. There are few number of works such as the IoT-related ontology SAREF[1]: Smart Appliance REFerence and iot-light[2]), which are supported by more general models. Although the vocabularies used in such ontologies are aligned with their generalized counterparts, the representation of the key concepts in sensor-related environment (such as sensor, action and observation) is limited [13].

Among the work in the literature, we can refer to SSN (Semantic Sensor Network) ontology [8] as an exception model with the main focus on designing a generic and reusable knowledge model for sensor-related environments. Relying on the upper level ontology DUL[3], the SSN ontology has taken a remarkable step towards the reusability of ontologies. In this paper, we propose a generic ontology for smart homes relying on SSN. Due to the SSN basis of our proposed ontology, instead of going to the details of the SSN-inspired works [9,10] that only represent a limited aspect of our domain of interest, we concentrate on SSN to see which parts are suitable to be used as a building block of a comprehensive and domain-independent smart home ontology.

Regarding the design pattern approach, the ODP techniques underpinning our our approach make it similar to the work proposed in [12], which introduces a generic Stimulus-Sensor-Observation ontology design pattern for representing observation-based data on the Semantic Web. Moreover, the focus of the work presented in [13] is on designing a core domain IoT ontology and proposes a reasonable categorization of high level concepts. What makes our approach different, however, is first its more comprehensive coverage of concepts discussed in the following section, and also the provided representational details of the aforementioned aspects of a smart home environment. We motivate our work in relation to some shortcomings in SSN described in the next subsection.

### *1.2. SSN Representational Issues*

SSN as a generic ontology that includes the semantic aspects of sensor-related domains has been also used to model smart environments [9]. However, we have encountered a number of representational issues in SSN. These issues as well as our suggestions to address them are listed in the following:

1. In SSN differentiating between the class `SSN:Observation` and the class `SSN:Sensing` is not straightforward. Since representation of a sensing process can provide all the details related to an observation process, we found that the class `SSN:Sensing` is enough to be seen as a general class of all the possible sensing processes. Therefore, we have developed a pattern called **SmartHome_sensing** (see Section 3.1.6) which simplifies and at the same time provides sufficient representational features to model a sensing process in a smart home.

[1] https://sites.google.com/site/smartappliancesproject/ontologies
[2] http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite
[3] DOLCE Ultra Light: www.ontologydesignpatterns.org/ont/dul/DUL.owl

2. In the context of smart homes, defining a feature of interest as the combination of an object with one of its properties makes more sense, rather than only as either an object or an event (as in SSN) that can have properties. For instance, a feature of interest in a smart home can be *the temperature of the room* as a standalone concept, and not the *room* on its own, which in turn has *temperature* as one of its properties. The difference becomes more obvious when we know that there is also *illumination* as the second property of *the room*, which may not however be of interest to any sensing process. To provide a standalone definition of a feature of interest in a smart home we have designed a pattern called **SmartHome_FeatureOfInterest** (see Section 3.1.4).
3. Physical objects play a major role in recognizing activities and events in a smart home. Although the fact that SSN relies on DUL ensures a rich representation of physical objects, we still need a generic taxonomy whereby different types of objects (e.g., smart objects, mobile objects, etc) whose instances can be found in a smart home, are represented (see Section 3.1.9).
4. Each sensor environment is equipped with a sensor network. An ideal smart home is also assumed to be capable of addressing sensor failures and perform other kinds of network configurations in case of problems. The general representational details of a smart home sensor network has been captured in the pattern **SmartHome_Network** (see Section 3.1.8). By integrating this pattern that contains all the physical aspects of a sensor network with the object pattern, we enable the ontology to also represent robots as a set of sensors attached to mobile objects (e.g., the arm of the robot which is movable).
5. In a smart home, specifically when there are mobile agents such as robots, we need to also know the topology of objects, places and other spatial entities. The only available spatial relations that exist in SSN (borrowed from DUL) are `DUL:nearTo` and `DUL:overlaps`. An extension of both spatial and directional relations between different objects and places are needed. In order to provide the required geometrical representation in a smart home, we have designed a pattern called **SmartHome_Geometry** (see Section 3.1.2).
6. In a smart environment, where all the activities are assumed to be captured by sensors, the concept of an event needs to be more general in order to also represent activities inferred from detecting changes (events) in sensor data. These activities can either be directly or indirectly captured (inferred) from the sensor data, and consist of different types of events whose specific representations are missing in SSN. For this, we have developed a content ontology pattern **SmartHome_Event** (see Section 3.1.10) to provide the required representational details.
7. The disjoint relationship between the class `SSN:Observation` and the class `DUL:Event` is known as one of the reasons causing an interoperability issue in SSN. The work introduced in [11] aims to deal with this issue by proposing a new implementation of O&M[4] with fewer dependencies on upper level ontologies such as DUL. In our current work, however, we deal with this issue by specializing the links between the patterns **SmartHome_Event** (see Section 3.1.10) and **SmartHome_Sensing** (see Section 3.1.6), while still taking advantage of relying on upper level ontologies, which potentially allows us to make our ODPs as general as possible.

Due to the aforementioned issues, we decided to create a network of modules whose basis are extracted from SSN. Each module is represented in the form of a pattern, as general as possible with the least possible dependencies on the other patterns. Doing so helps us to realise the main links in the eventual ontology. We can consequently make a stable and at the same time flexible network of concepts that can be updated with the minimum change propagation on the entire ontology, and where individual patterns can also be used in isolation for some specific reasoning tasks (e.g., in order to avoid issues with reasoning complexity or clashed in the relations to foundational ontologies as mentioned earlier).

Before going to the details of the ontology, in the next section, we shortly introduce the project whose requirements led us to the development of the SmartHome ontology in the form of a pattern language. In this section we also introduce the technical parts of the project related to the representation model. In Section 3, we explain the

[4] http://www.opengeospatial.org/standards/om

10 ontology patterns (modules) used in SmartHome ontology. The SmartHome ontology, as the result of specializing the relations between modules, is illustrated in Section 4. A discussion on our approach is given in Section 5. The paper is concluded by giving remarks concerning future developments.

## 2. Use Case

In order to set the stage and explain the background of our work, we will here briefly describe the project where the pattern language was developed. We also list some requirements of the SmartHome ontology specifically needed for that project, and introduce a reasoning example that will be used throughout the paper to exemplify the use of the modules.

### *2.1. Ecare@Home Project*

The E-Care@Home project aims at providing a comprehensive IoT-based healthcare system, including state of the art communication protocols and high-level analysis of data from various types of sensors, combined with information from personal health records, and other background information, both generic and specific to a person. The main scenario is that of an elderly person who has some specific needs and potential medical conditions, but is still living at home. In order to increase the safety of the person, and reduce the frequency of appointments needed at a care facility, the patient and caregivers have agreed to fit the patients home with some sensors and a communication device, such as a tablet with a specific application installed. The challenge is to integrate and reason over all the information both from the sensors and the medical records at once, and derive the most likely conclusion, e.g., the current situation that the patient is in, the cause of some events, and the best action for the system to take next. This is quite a typical scenario for sensor-base monitoring systems, hence, it has enabled us to generalise our specific requirements and provide a solution that we believe is highly reusable by other systems, regardless of the domain where such situation awareness and monitoring is required.

### *2.2. Conceptual Aspects and Competency Questions*

During the requirements analysis process, we considered a number of aspects of smart homes to be covered in the ontologies. A set of high-level competency questions (CQs) for each aspect have also been determined (only examples are given here due to space restrictions). The detailed CQs for each of the patterns can be found inside the ODP itself and on its page in the ODP portal:

- **Sensors:** To what object is a sensor attached? What is the location of the object, and what does the sensor measure? Can the sensor or its holding object move?
- **Locations:** What are the locations adjacent to a specific location, and what activities can be carried out in this location?
- **Objects:** What are the (static and dynamic) properties of a given object?
- **Agents:** What are the possible activities of an agent? Can the agent be targeted by sensors?
- **Observations:** What can be observed by a certain sensor? What complex events can be inferred from simple observations?
- **Activities:** What activities can be inferred based on sensor observations and other activities, for a certain agent in a certain location?

One important task of our system, which is also typical to similar types of systems having been frequently reported in literature(c.f. [1]), is to identify the activities of a person in the environment such as “sleeping”, “watching TV”, “cooking” etc. In addition to being able to identify an activity, we are also interested in the time and location where the activity is carried out. By selecting this quite generic task as an example, we ensure that the reusability of the modules will also be illustrated through our examples.

#### *2.2.1. A Sample Scenario*

Throughout this paper we will use the following example to illustrate our modules:

Let us assume we are monitoring a patient with severe COPD (Chronic Obstructive Pulmonary Disease). Since one effect of the disease is lung function reduction, patients tend to have a hard time to remain active. A reduced level of activity in their daily lives may be an indication of a worsening of the condition. The level of activity is furthermore seen as an important contextual background information for interpreting readings from medical sensors and self assessments, such as perceived breathlessness or oxygen saturation. Therefore one task of the E-Care@Home system is to create a time-line of the patient’s daily activities. One possible (simplified) example of such an inference, based on sensor

observations could be that the patient is watching TV as long as the person is seated on the couch in the living room and the TV is on. For performing this inference, we need at least two *sensors* attached to *objects* located in the *living room*: one that registers the coach occupancy and one that records the on/off status of the TV. Moreover, we need information about the patient, as an *actor* (or agent), who's activities can be observed through *sensing* processes implemented by these sensors. We also need several layers of abstraction in terms of observed events, i.e., both low-level manifestations, such as that the TV is on, and complex events that are composed of (sequences or sets) of such manifestations, such as the notion of watching TV, as well as reasoning mechanisms to derive the latter from the former.

## 3. SmartHome Ontology – Overview

In the previous section, we discussed the existing ontologies that have inspired our work, and the reasons why none of them cover all our requirements, and in addition they all lack the desirable properties of a network of modules. In this section, we briefly provide an overview of the overall ontology network, i.e. the pattern language, that we propose as a solution to this problem. Figure 1 represents an abstract overview of the overall ontology in terms of its modules and their relations. The figure is somewhat simplified in order to be more readable, e.g., most inverse relations have been omitted as well as some of the alignments to external classes and properties. The connections between classes and arbitrary properties in the figure are either based on domain and range restrictions, or restrictions on the class itself. In the following section we then go into details of the individual modules and their axiomatisation.

### 3.1. Patterns

The SmartHome ontology[5] is composed of 10 ontology modules which are publicly available. Each module is representing a single principal feature of a smart home. The requirements for the modules of the SmartHome ontology are formalized based on CQs and reasoning requirements, derived from the high-level requirements presented earlier. As we will see in the following subsections, most of the modules are extending SSN, which in turn has been designed based on DUL. The name of the concepts and relations taken from the aforementioned ontologies are represented in the form of hyper-links referring to the OWL definition of the concepts.

Some patterns are further explained with some examples after their DL notations. For the sake of declarativity, these examples are either in the form of a unary predicate (indicating class instantiation) or a binary predicate (indicating object properties).

#### 3.1.1. SmartHome_TimeInterval Pattern

[6] The pattern **SmartHome_TimeInterval** represents time in the form of time intervals between any two timestamps through an observation process. The timestamps indicating the boundary of an interval are usually representing an instance of `xsd:dateTime` type. However, there are other situations where we need to relatively indicate a temporal distance in the form of the number of timestamps, regardless of the date/time when each of these timestamps is measured.

In order to differentiate these two types of intervals, the **SmartHome_TimeInterval** pattern provides two classes *SmartHomeTimeInterval* and *SmartHomeTemporalDistance* that allow us to answer queries about the temporal distance between any two specific timestamps dedicated to different events.

*SmartHomeTimeInterval* is subsumed by the class `DUL:TimeInterval` and indicates two time stamp values[7] as the boundary of the interval. This class can be used in different occasions such as capturing an event occurred within a specific time interval (e.g., *TV was on between 11:00 a.m and 11:30 a.m*).

$$\begin{aligned} SmartHomeTimeInterval \sqsubseteq\ & \texttt{DUL:TimeInterval} \sqcap \\ & = 1\ hasLowerTimeStampValue.(\texttt{xsd:dateTime}) \\ & = 1\ hasUpperTimeStampValue.(\texttt{xsd:dateTime}) \end{aligned} \quad (1)$$

*SmartHomeTemporalDistance* is the second class of this pattern and likewise subsumed by the class `DUL:TimeInterval`. An instance of this class is meant to indicate a relative time distance between any two different events. This concept can be used in the definition of events whose preconditions need to be captured within a specific time distance regardless of their real date/time that

[5] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome

[6] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_TimeInterval

[7] It is an OWL notation and not a standard DL notation.

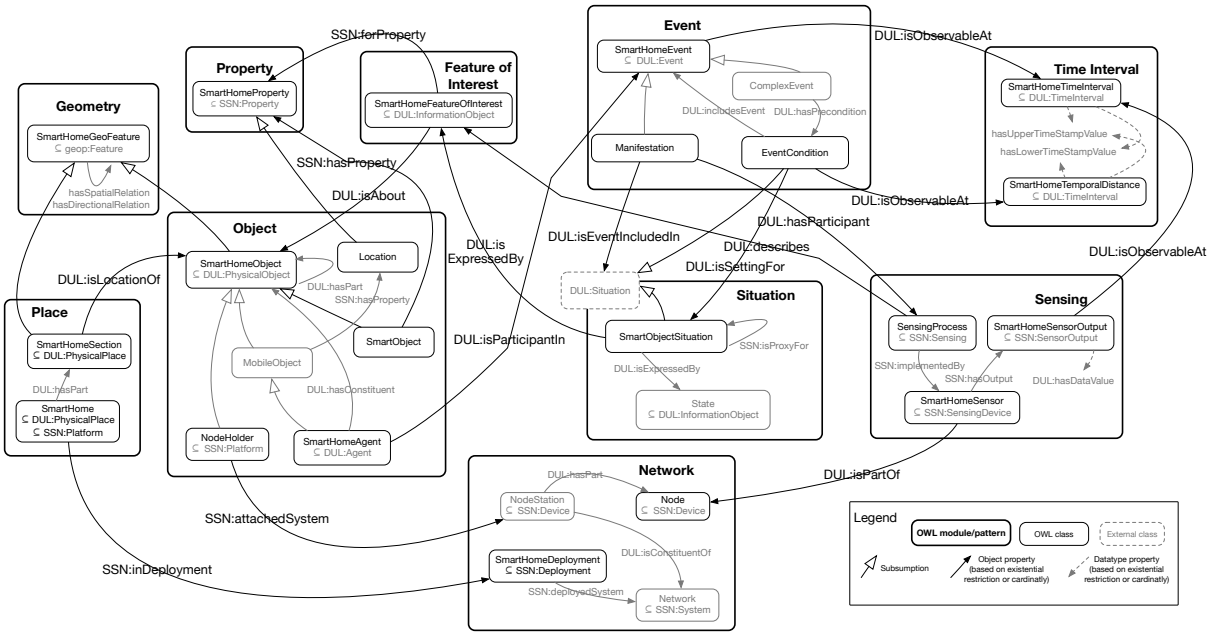


Fig. 1. Overview of the Smart Home Ontology

they occur (e.g., *eating usually occurs between 1-10 minutes after cooking.*). What makes the definition of *SmartHomeTemporalDistance* different from the definition of *SmartHomeTimeInterval* is the range type of their data properties:

$$\begin{aligned} SmartHomeTemporalDistance \sqsubseteq\ & \texttt{DUL:TimeInterval} \sqcap \\ & = 1\ hasLowerTimeStampValue.(\texttt{xsd:long}) \\ & = 1\ hasUpperTimeStampValue.(\texttt{xsd:long}) \end{aligned} \tag{2}$$

*hasLowerTimeStampValue* is subsumed by the data property `DUL:hasDataValue` and indicates a value of either type `xsd:dateTime` or type `xsd:long` as the timestamp of the lower bound.

*hasUpperTimeStampValue* is likewise subsumed by the data property `DUL:hasDataValue` and indicates a value of either type `xsd:dateTime` or type `xsd:long` as the timestamp of the upper bound.

### 3.1.2. SmartHome_Geometry Pattern

[8] Apart from the temporal aspect of a smart environment, the representational model needs to also cover the spatial aspects of entities (e.g., objects, rooms, etc) occupying space. For this, we have extended the GeoSPARQL ontology [14] to provide a basis for the representation of both the geometry of entities and the topological and directional relations between any pair of geometrical objects. For instance, there are many situations where we need to localize objects relative to the physical position of the user. For this, encoding the spatial relations between entities are essential (e.g., *the kitchen is connected to the living-room*, *the living room is at north of the bedroom*, or *the bed is close to the left wall.*).

The OGC GeoSPARQL standard defines an adequate vocabulary for representing geospatial data enabling qualitative spatial reasoning based on geometrical computations. The extension that we made upon GeoSPARQL is given in the following:

*SmartHomeGeoFeature* is subsumed by the class `geop:Feature`. According to GeoSPARQL, a feature represents a spatial object that has a geometry. We constrain the definition by saying that each feature is also in a spatial and directional relations with at least one another feature:

$$\begin{aligned} SmartHomeGeoFeature \sqsubseteq\ & \texttt{geop:Feature} \sqcap \\ & \exists\ \texttt{geop:hasGeometry.geop:Geometry} \\ & \exists\ hasSpatialRelation.SmartHomeGeoFeature \\ & \exists\ hasDirectionalRelation.SmartHomeGeoFeature \end{aligned} \tag{3}$$

[8] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_Geometry

*hasSpatialRelation* is an object property subsuming all the basic RCC-8 [15] topological relations (e.g., `geop:sfContains` $\sqsubseteq$ *hasSpatialRelation*, `geop:sfOverlaps` $\sqsubseteq$ *hasSpatialRelation*, etc.). The 8 directional relations (e.g., *northOf*, *northEastOf*, etc) are also represented as the sub properties of the object property *hasDirectionalRelation*:

$$hasDirectionalRelation \sqsubset hasSpatialRelation \quad (4)$$
$$\top \sqsubseteq \forall\ hasSpatialRelation^{-}.SmartHomeGeoFeature$$
$$\top \sqsubseteq \forall\ hasSpatialRelation.SmartHomeGeoFeature$$
$$\top \sqsubseteq \forall\ hasDirectionalRelation^{-}.SmartHomeGeoFeature$$
$$\top \sqsubseteq \forall\ hasDirectionalRelation.SmartHomeGeoFeature$$

### 3.1.3. SmartHome_Property Pattern

[9] Each (physical) object in a smart environment is defined based on its properties. These properties vary depending on the category of objects (e.g., mobile or non-mobile objects), and also the measurement criteria (e.g., location of objects) based on which the observation process is conducted. The pattern **SmartHome_Property** is used to more tightly couple the representation of physical objects with some features or properties which are measurable by sensors. For instance, pressure on the surface of a couch is considered as the property of the couch because it can be measured by a pressure sensor. Another example is the location of a mobile object such as a chair whose location can be tracked using RFID sensors covering the floor and also under the feet of the chair.

We assume that objects in the context of smart environments are represented as subclasses of the class `DUL:PhysicalObject` (see Section 3.1.9). Under this assumption, a property of an object indicates a feature of the object (e.g., the *pressure* of the couch, or the *location* of the chair) and is defined within the class `SSN:Property`. This pattern also borrows the two object properties: `SSN:hasProperty` and `SSN:isPropertyOf` from `SSN`.

*SmartHomeProperty* is subsumed by the class `SSN:Property` and defines a measurable property of a physical object as follows:

$$SmartHomeProperty \sqsubseteq \texttt{SSN:Property} \sqcap \quad (5)$$
$$\exists\ \texttt{SSN:isPropertyOf.DUL:PhysicalObject}$$

[9] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_Property

Example:

```
SmartHomeProperty(pressure).
SmartHomeProperty(luminosity).
```

The relations between *SmartHomeProperty* and the instances of the class `DUL:PhysicalObject` are provided by the object property `SSN:isPropertyOf`.

### 3.1.4. SmartHome_FeatureOfInterest Pattern

[10] A feature of interest is commonly referring to a concept deemed as the interest of an observation process (e.g., "temperature of the oven"). There are many situations where a feature of interest is seen as merely a piece of information without mentioning its numerical (e.g., 70 °C) or even symbolic (e.g., warm) values. We, therefore, specialize the class `DUL:InformationObject`[11] so as to represent the instances of the class *SmartHomeFeatureOfInterest* as pieces of information objects.

Since this pattern is about the properties of objects, the **SmartHome_Property** pattern is also imported to contribute in the definition of the axioms given in the following:

*SmartHomeFeatureOfInterest* is defined as a subsumption of the class `DUL:InformationObject`. The class definition is completed by adding two new axioms which relate the class to the `DUL:PhysicalObject` class via `DUL:isAbout` object-property and also to the class `SmartHomeProperty` from the **SmartHome_Property** pattern via `SSN:forProperty`.

$$SmartHomeFeatureOfInterest \sqsubseteq \texttt{DUL:InformationObject} \sqcap \quad (6)$$
$$\exists\ \texttt{DUL:isAbout.DUL:PhysicalObject} \sqcap$$
$$\exists\ \texttt{SSN:forProperty.SmartHomeProperty}$$

[10] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_FeatureOfInterest

[11] According to DUL, an information object is defined as a piece of information independently from how it is concretely realized.

```
Example:
SmartHomeFeatureOfInterest(foi_tv_lum).
SmartHomeFeatureOfInterest(foi_couch_presure).
PhysicalObject(tv_1).
PhysicalObject(couch_1).
isAbout(foi_tv_lum, tv_1).
forProperty(foi_tv_lum, luminosity).
isAbout(foi_couch_presure, couch_1).
forProperty(foi_couch_presure, pressure).
```

Therefore, an instance of the class *SmartHomeFeatureOfInterest* indicates an object and its property monitored in a (sensing) process.

### *3.1.5. SmartHome_Situation Pattern*

[12] A feature of interest can be declaratively expressed and indicate a specific situation. More specifically, a "situation" refers to a specific *state* of a "feature of interest" (e.g., the temperature of the living room is *warm*). Although states are usually time dependent, we decided to keep the representation of a situation as abstract as possible for the sake of generality. The concept of situation can be augmented with the concept of time in other patterns such as event-related patterns which are associated with temporal properties (see Section 3.1.10).

Since a situation is about a feature of interest, the pattern **SmartHome_Situation** relies on the pattern **SmartHome_FeatureOfInterest** explained in Section 3.1.4. **SmartHome_Situation** contains two classes *SmartObjectSituation* and *State* where the individuals of the former are expressed by the individuals of the latter class.

Two situations can be regarded as equivalent from an observation process perspective, despite the differences between the details of their involved objects and the properties. For instance, one can say that the two situations "*stove is on*" (which indicates that the electric current switch of the stove is on) and "*the LED of the stove is on*" represent the same fact about the stove and therefore can be used interchangeably.

*SmartObjectSituation* is subsumed by the class `DUL:Situation`. A situation in a smart home can be more specialized and expressed by a feature of interest (e.g., temperature of living room) and its state (e.g., warm). Therefore, the definition of the class also includes two axioms that determine the relations between a feature of interest and its relevant states. These two axioms rely on the object property `DUL:isExpressedBy`. Furthermore, as said before, a situation can be equivalent to (proxy for) another situation from the observation process perspective. Therefore, the DL notations of the class *SmartObjectSituation* is as follows:

$$\begin{aligned} SmartObjectSituation \sqsubseteq\ & \texttt{DUL:Situation} \sqcap \\ & \exists\ \texttt{DUL:isExpressedBy}.SmartHomeFeatureOfInterest \sqcap \\ & \exists\ \texttt{DUL:isExpressedBy}.State \sqcap \\ & \exists\ \texttt{SSN:isProxyFor}.SmartObjectSituation \end{aligned} \tag{7}$$

*State* is also a class whose individuals are assumed to declaratively express a feature of interest regardless of how this expression is realized. We, therefore, define the class *State* as a subclass of the generic class `DUL:InformationObject` which according to DUL is able to *express* instances of the class `DUL:Situation`. However, for the sake of specificity, we limit the definition of the class by saying that it also expresses a specialized situation i.e. a *SmartObjectSituation*:

$$\begin{aligned} State \sqsubseteq\ & \texttt{DUL:InformationObject} \sqcap \\ & \exists\ \texttt{DUL:expresses}.SmartObjectSituation \end{aligned} \tag{8}$$

```
Example:
SmartObjectSituation(situ_tv_lum_on).
SmartObjectSituation(situ_couch_pressed).
State(on).
State(pressed).
isExpressedBy(situ_tv_lum_on, foi_tv_lum).
isExpressedBy(situ_tv_lum_on, on).
isExpressedBy(situ_couch_pressed, foi_couch_presure).
isExpressedBy(situ_couch_pressed, pressed).
```

### *3.1.6. SmartHome_Sensing Pattern*

[13] A sensing process is simply defined as the process of monitoring a feature of interest using a sensing device. Since the pattern **SmartHome_Sensing** is also about a feature of interest, we therefore have used the pattern **SmartHome_FeatureOfInterest** in the definition of its classes. In order to be established, a sensing process (as a method[14]) needs at least a phenomenon to sense (i.e. a feature of interest) or more specifically to describe its state, and also

[12] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_Situation

[13] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_Sensing

[14] By method, we refer to the class `DUL:Method` $\sqsubseteq$ `SSN:Process` $\sqsubseteq$ `SSN:Sensing`

a sensing device whose outputs will be regarded as the result of the process. According to these preliminary ingredients inspired from SSN, the pattern **SmartHome_Sensing** has been designed based on the following three main classes:

*SensingProcess* is a class subsumed by the class SSN:Sensing. It is worth mentioning that each sensing process can be implemented by several sensors, each has its own output (i.e. raw sensor data). Furthermore, a sensing process is seen as a process describing a feature of interest by measuring its state.In the following the definition of the class *SenisngProcess* represents all the required relations:

$$\begin{aligned} \mathit{SensingProcess} \sqsubseteq\ & \texttt{SSN:Sensing} \sqcap \\ & \exists\ \texttt{SSN:implementedBy}.\mathit{SmartHomeSensor} \sqcap \\ & \exists\ \texttt{DUL:describes.SmartHomeFeatureOfInterest} \end{aligned} \tag{9}$$

As we will see in Section 3.1.10, the relation between an event and the sensing process supposed to capture it, is indirectly provided via the two class of *SmartHomeFeatureOfInterest* and *SmartObjectSituation* (see Figure. 1).

*SmartHomeSensor* is subsumed by the class SSN:SensingDevice. A sensor in the context of a smart home is a sensing device that can be as such a sub-part of another device (e.g., a data sending module in a network). Each sensor is involved in (implements) a sensing process and produces numerical outputs (i.e. raw sensor data) as the result of this process. The following DL notations reflect the aforementioned details of the class *SmartHomeSensor*:

$$\begin{aligned} \mathit{SmartHomeSensor} \sqsubseteq\ & \texttt{SSN:SensingDevice} \sqcap \\ & \exists\ \texttt{SSN:hasOutput}.\mathit{SmartHomeSensorOutput} \sqcap \\ & \exists\ \texttt{SSN:implements}.\mathit{SensingProcess} \sqcap \\ & \exists\ \texttt{DUL:isPartOf.SSN:Device} \end{aligned} \tag{10}$$

Example:

```
Sensor(luminosity_sensor_1).
Sensor(pressure_sensor_2).
SensingProcess(process_1).
SensingProcess(process_2).
implementedBy(process_1, luminosity_sensor_1).
implementedBy(process_2, pressure_sensor_2).
describes(process_1, foi_tv_lum).
describes(process_2, foi_couch_presure).
```

*SmartHomeSensorOutput* is subsumed by the class SSN:SensorOutput. We expect to gain from the definition of sensor output, something about the sensor, the output value and the timestamp at which the value is captured.

$$\begin{aligned} \mathit{SmartHomeSensorOutput} \sqsubseteq\ & \texttt{SSN:SensorOutput} \sqcap \\ & \exists\ \texttt{DUL:isObservableAt.DUL:TimeInterval} \sqcap \\ & \exists\ \texttt{SSN:isProducedBy}.\mathit{SmartHomeSensor} \sqcap \\ & =\ 1\ \texttt{DUL:hasDataValue.(xsd:long)} \end{aligned} \tag{11}$$

### 3.1.7. SmartHome_Place Pattern

[15]

The meaning of a place in the context of smart home is twofold. First, by a place we mean the entire smart environment which holds the deployment of a sensor network and might also be composed of several sections. The second meaning of a place refers to each section of the main place with a specific identity that can be as such seen as a location for different objects. Given this preliminary definition, the pattern **SmartHome_Place** defines a place as a specialization of the class DUL:PhysicalPlace with the following details:

*SmartHome* represents the first type of a smart home place (i.e. the entire environment). As mentioned above, this type of place is a DUL:PhysicalPlace that is also assumed to hold a deployment process e.g., a sensor network. This class, therefore, has to also be subsumed by the class DUL:Platform, which according to SSN, provides a link to a deployment process (i.e. an instance of the class SSN:Deployment):

$$\begin{aligned} \mathit{SmartHome} \sqsubseteq\ & \texttt{DUL:PhysicalPlace} \sqcap \\ & \texttt{SSN:Platform} \sqcap \\ & \exists\ \texttt{SSN:inDeployment.SSN:Deployment} \sqcap \\ & \exists\ \texttt{DUL:hasPart}.\mathit{SmartHomeSection} \end{aligned} \tag{12}$$

*SmartHomeSection* represents spatial sections as parts of a smart home. Each section defines a physical place that can accommodate different objects. Furthermore, each spatial section in a smart home has a geometry and therefore, can be in spatial relations with the other sections. In order to reflect such properties for the class SmartHomeSection, we specialize the definition by adding a subsumption

[15] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_Place

relation with the class *SmartHomeGeoFeature* defined in the **SmartHome_Geometry** pattern:

$$\begin{aligned} SmartHomeSection \sqsubseteq\ & \texttt{DUL:PhysicalPlace} \sqcap \\ & SmartHomeGeoFeature \sqcap \\ & \exists\ \texttt{DUL:isLocationOf.DUL:PhysicalObject} \sqcap \\ & \exists\ \texttt{DUL:isPartOf.}SmartHome \end{aligned} \tag{13}$$

```
Example:
SmartHome(the_smart_home).
SmartHomeSection(livingroom_1).
isPartOf(livingroom_1, the_smart_home).
isLocationOf(livingroom_1, tv_1).
isLocationOf(livingroom_1, couch_1).
```

### *3.1.8. SmartHome_Network Pattern*

[16] A network in a smart environment is defined as a system containing different types of devices such as nodes and node stations. By node, we mean a communication module that indicates either a sending or a receiving data module in a network. Each node depending on its type can be a part of a node station representing another type of device that contributes in establishing a network. Each node station contains a node along with other things including a sensor, power supplies, batteries etc.

The whole process of a network set-up regardless of its exact technical details is represented by a non-physical concept called deployment. The pattern **SmartHome_Network** unifies the representation of home automation installations variety of which can be found in different systems.

*SmartHomeDeployment* extends the class `SSN:Deployment` and indicates a platform (e.g., a smart home) upon which a system (e.g., a network) is deployed[17]:

$$\begin{aligned} SmartHomeDeployment \sqsubseteq\ & \texttt{SSN:Deployment} \sqcap \\ & \exists\ \texttt{SSN:deployedOnPlatform.SSN:Platform} \sqcap \\ & \exists\ \texttt{SSN:deployedSystem.}Network \end{aligned} \tag{14}$$

*Network* is subsumed by the class `SSN:System`. This class is further specialized by relating it to a deployment process as well as to its constituents:

$$\begin{aligned} Network \sqsubseteq\ & \texttt{SSN:System} \sqcap \\ & \exists\ \texttt{DUL:hasConstituent.}NodeStation \sqcap \\ & \exists\ \texttt{SSN:hasDeployment.}SmartHomeDeployment \end{aligned} \tag{15}$$

*NodeStation* represents a `SSN:Device` (either a *SenderNodeStation* or a *ReceiverNodeStation*), which is located on a platform (e.g., a node holder) in the environment and can contain a number of nodes[18]:

$$\begin{aligned} NodeStation \sqsubseteq\ & \texttt{SSN:Device} \sqcap \\ & \exists\ \texttt{DUL:hasPart.}Node \sqcap \\ & \exists\ \texttt{DUL:isConstituent.}Network \sqcap \\ & \exists\ \texttt{SSN:onPlatform.SSN:Platform} \end{aligned} \tag{16}$$

*Node* likewise represents a `SSN:Device`, either a *DataReceiverNode* or a *DataSenderNode*, that is included in a node station:

$$\begin{aligned} Node \sqsubseteq\ & \texttt{SSN:Device} \sqcap \\ & \exists\ \texttt{DUL:isPartOf.}NodeStation \end{aligned} \tag{17}$$

*DataReceiverNode* as its name indicates, models a node (as part of a node station) which receives data coming from sensors (or more specifically from senders modules):

$$\begin{aligned} DataReceiverNode \sqsubseteq\ & Node \sqcap \\ & \exists\ \texttt{DUL:isPartOf.}ReceiverNodeStation \sqcap \\ & \exists\ receivesDataFrom.DataSenderNode \end{aligned} \tag{18}$$

*DataSenderNode* also models a node which as a part of a node station sends sensor data (to a receiver):

$$\begin{aligned} DataSenderNode \sqsubseteq\ & Node \sqcap \\ & \exists\ \texttt{DUL:isPartOf.}SenderNodeStation \sqcap \\ & \exists\ sendsDataTo.DataReceiverNode \end{aligned} \tag{19}$$

*ReceiverNodeStation* defines a node station which holds a data receiver node as its part:

$$\begin{aligned} ReceiverNodeStation \sqsubseteq\ & NodeStation \sqcap \\ & \exists\ \texttt{DUL:hasPart.}DataReceiverNode \end{aligned} \tag{20}$$

[16] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_Network

[17] `SSN:deployedOnPlatform` is the inverse property of `SSN:inDeployment`

[18] `SSN:onPlatform` is the inverse property of `SSN:attachedSystem`

*SenderNodeStation* likewise represents a node station which is assumed to contain both a data sender node and also sensing devices (sensors):

$$SenderNodeStation \sqsubseteq NodeStation \sqcap \exists \texttt{DUL:hasPart}.DataSenderNode \sqcap \exists \texttt{DUL:hasPart.SSN:SensingDevice} \quad (21)$$

*receivesDataFrom* is an object property providing the relation between a DataReceiverNode and a DataSenderNode. It is also the inverse of the object property *sendsDataTo*:

$$\top \sqsubseteq \forall\ receivesDataFrom.DataSenderNode \quad (22)$$
$$\top \sqsubseteq \forall\ receivesDataFrom^{-}.DataReceiverNode$$
$$receivesDataFrom \equiv sendsDataTo^{-}$$

### 3.1.9. SmartHome_Object Pattern

[19]

The pattern **SmartHome_Object** allows us to define objects based on their important features or abilities in the context of smart environments. The class `DUL:PhysicalObject` provides a suitable representational basis for the objects' taxonomy, which we have categorized into two types of *SmartObject* and *NodeHolder*. By *SmartObject* we refer to those objects that are the interest of an observation process. Due to the the usual difficulties of installing sensors in a smart home, it is common to use some other objects (i.e. *NodeHolder*s) to hold sensors. This separation provided by this pattern is specifically useful for other computational modules such as a configuration planner one of whose tasks is checking the status/functionality of sensors by sending a robot to their locations.

Each smart object has at least a property (to be monitored) represented by the pattern **SmartHome_Property**, and that is why this pattern has been used in the definition of the classes of the object pattern. Another categorization of smart objects that has been considered in the object pattern, is about their mobility. An objects is considered as mobile only if its location as one of its properties, can change.

In order to also be able to reflect the spatial relations between objects (e.g., the "fridge is connected to the cabinet"), or between an object and a place where it is located (e.g., "the bed is located at the left side of the bedroom"), it is required to define the *SmartHomeObject* also as a *SmartHomeGeoFeature* defined in the pattern **SmartHome_Geometry** (see Section 3.1.2).

In the following, we represent the DL notations of each object type along with their properties.

*SmartHomeObject* is subsumed by the class `DUL:PhysicalObject`. The class definition is specialized according to the aforementioned features of a smart home object as follows:

$$SmartHomeObject \sqsubseteq \texttt{DUL:PhysicalObject} \sqcap SmartHomeGeoFeature \sqcap \exists \texttt{DUL:hasLocation.DUL:PhysicalPlace} \quad (23)$$

*SmartObject* is a *SmartHomeObject* that, as said above, represents an object having a property which is the interest of an observation process. For this to be represented, we have included the pattern **SmartHome_Property** in the **SmartHome_Object** pattern:

$$SmartObject \sqsubseteq SmartHomeObject \sqcap \exists \texttt{SSN:hasProperty.SmartHomeProperty} \quad (24)$$

*NodeHolder* is also a *SmartHomeObject* representing the objects that act as a platform (or holder) for a device (a node) in a sensor network. In this way, we can differentiate between the objects contributing in the set-up of a smart home, namely objects which are the interest of the observation, and the objects which, as the holder of sensors, are used for the sensor localization process.

$$NodeHolder \sqsubseteq SmartHomeObject \sqcap \texttt{SSN:Platform} \sqcap \exists \texttt{SSN:attachedSystem.SSN:Device} \quad (25)$$

*MobileObject* defines a movable object that can be found at different places in the environment. In other words, there is a permanent property of interest called "Location" for mobile objects:

$$MobileObject \sqsubseteq SmartHomeObject \sqcap \exists \texttt{SSN:hasProperty}.Location \quad (26)$$

where:

$$Location \sqsubseteq SmartHomeProperty \sqcap \exists \texttt{SSN:isPropertyOf}.MobileObject$$

[19] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_Object

The specialized definition of the class SmartHomeProperty from the **SmartHome_Property** pattern is also given in the following:

$$\texttt{SmartHomeProperty} \sqsubseteq \exists\, \texttt{DUL:isPropertyOf}.SmartObject \tag{27}$$

*SmartHomeAgent:* The class *MobileObject* can be further specialized and form another type of objects that are able to be proactive and participate in some events. The class *SmartHomeAgent* subsumed by the class `DUL:Agent` allows us to represents smart home objects such as inhabitant persons or pets, that can be involved in an activity or an event (e.g., a cat at home is defined as an agent because it is able to be involved in the activity of "sitting on the couch", or likewise a person is also an agent at home as he/she is often involved in various activities such as "watching TV", etc.). Each smart home agent can own, as its constituent, at least one *SmartHomeObject* involved in a sensing process. A good example of such constituents are the body parts of the agent, which as a *SmartObject* (e.g., the *heart* of a person) or a *NodeHolder* (e.g., the *arm* of the person attached on it is a pressure sensor) are involved in a sensing process. A constituent of a smart home agent, in general, is referring to objects, location of which depends on the location of the agent. More specifically, a constituent can express other objects physically attached to the agent. However, a constituent does not need to be permanently attached to the agent. For instance, a chair might be deemed as a constituent as long as it is held by the agent.

$$SmartHomeAgent \sqsubseteq \texttt{DUL:Agent} \sqcap MobileObject \sqcap \exists\, \texttt{DUL:hasConstituent}.SmartHomeObject \sqcap \exists\, \texttt{DUL:isParticipantIn}.\texttt{DUL:Event}$$

Example:

```
SmartObject(tv_1).
SmartObject(couch_1).
SmartHomeAgent(john).
SmartObject(heart_1).
NodeHolder(arm_1).
SmartHomeProperty(rate).
SmartHomeFeatureOfInterest(foi_heart_rate).
isAbout(foi_heart_rate, heart_1).
forProperty(foi_heart_rate, rate).
hasConstituent(john, heart_1).
hasConstituent(john, arm_1).
```

Given the definition of the mobile objects, one can ask about a situation where a mobile object moves outside of a smart home. For instance, the inhabitant person is holding his/her chair and going outside. In order to infer the new location of the chair, it is only required to infer that the person has left the place (i.e. "there is zero inhabitant inside"). In this way, we can therefore know that the chair held by the person is not inside either, as it is temporarily one of his/her *constituent*[20].

### *3.1.10. SmartHome_Event Pattern*

[21] From the observation process viewpoint, an event can be seen as either a manifestation or a complex event, where the former refers to those events that can be directly captured from sensor data and represent the occurrence of a smart home situation through a sensing process. However, the latter event type, as its name indicates, represents more complicated events whose occurrence depends on several preconditions [10]. Each precondition as such represents a specific situation assumed to be observed within an interval with a specific temporal distance to the event's occurrence time.

The class `DUL:Event` is suitable enough to be as the basis of the definition of a smart home event. Each smart home event is expected to be observable at/within a `DUL:TimeInterval`. Furthermore, as mentioned in Section 3.1.9, an event can have an agent (a proactive object) as its participant.

*SmartHomeEvent* as a general event class in the context of smart home is subsumed by the class `DUL:Event`. The definition is more specialized by including the participants of the event as well as the TimeInterval during which the event occurs:

$$SmartHomeEvent \sqsubseteq \texttt{DUL:Event} \sqcap \exists\, \texttt{DUL:hasParticipant}.\texttt{DUL:Agent} \sqcap \exists\, \texttt{DUL:isObservableAt}.\texttt{DUL:TimeInterval} \tag{28}$$

*Manifestation* as a specialized version of the class *SmartHomeEvent* indicates a situation (more specifically a SmartHomeSituation) directly captured from sensor data:

[20] We are currently working on a knowledge-driven approach to keep the current number of agents at home. The details of this work is out of the scope of this paper.

[21] http://ontologydesignpatterns.org/wiki/Submissions:SmartHome_Event

$$Manifestation \sqsubseteq SmartHomeEvent \sqcap \exists \texttt{DUL:isEventIncludedIn.DUL:Situation} \quad (29)$$

*ComplexEvent* is also as a specialized version of the class *SmartHomeEvent* and represents an event whose occurrence depends on capturing a number of preconditions represented as situations:

$$ComplexEvent \sqsubseteq SmartHomeEvent \sqcap \exists \texttt{DUL:hasPrecondition.}EventCondition \quad (30)$$

*EventCondition* as a `DUL:Situation` represents preconditions of a complex event (in the form of a situation, or more specifically a SmartHomeSituation) which are needed to be captured within a specific temporal distance from the timestamp of the complex event.

$$EventCondition \sqsubseteq \texttt{DUL:Situation} \sqcap \exists \texttt{DUL:isPreconditionOf.}ComplexEvent \sqcap \exists \texttt{DUL:isObservableAt.DUL:TimeInterval} \sqcap \exists \texttt{DUL:isSettingFor.DUL:Situation} \quad (31)$$

Example:

WatchingTV ⊑ ComplexEvent

```
WatchingTV(watching_tv).
EventCondition(condition_1).
EventCondition(condition_2).
hasPrecondition(watching_tv, condition_1).
hasPrecondition(watching_tv, condition_2).

SmartHomeTemporalDistance(temp_distance_1).
hasLowerTimeStampValue(temp_distance_1, 10).
hasUpperTimeStampValue(temp_distance_1, 0).
isObservableAt(condition_1, temp_distance_1).

SmartHomeTemporalDistance(temp_distance_2).
hasLowerTimeStampValue(temp_distance_2, 0).
hasUpperTimeStampValue(temp_distance_2, 0).
isObservableAt(condition_2, temp_distance_2).

isSettingFor(condition_1, situ_couch_pressed).
isSettingFor(condition_2, situ_tv_lum_on).
```

## 4. Ontology Network and its Application

In this section, we explain how the SmartHome ontology is formed as the result of importing all the 10 modules. The connection between each pair of modules is accomplished by specializing the definition of concepts in each module and then linking them together. For instance, the class *Deployment* used in the pattern **SmartHome_Network** is specialized to the class *SmartHomeDeployment* (see Figure 1), and then forms the link to the **SmartHome_Place** pattern via the `SSN:inDeployment` property: "$\exists$ `SSN:inDeployment.`*SmartHomeDeployment*". All the specialized relations between modules illustrated in Figure 1, are also listed in Table 1.

In E-Care@home the ontology has been used as the representation model of the data that is gathered from a deployment lab in Örebro University called PEIS-Home. PEIS-Home provides functional facilities for the research development including Ambient Intelligence (AMI) solutions. The SmartHome ontology was first populated with the static information about the PEIS-Home set-up which totally resulted in 172 specialisations (i.e. subclasses) of the ODP classes, and 203 individuals. Given this instantiated ontology, the observation process is then able to feed the ontology with the instances of manifestations corresponding to the changes detected in sensor outputs. For instance, monitoring a person doing different sorts of activities (such as cooking, watching TV, etc.) in the PEIS-Home for 5 minutes resulted in about 200 additional individuals, describing the situations captured from the environment. These individuals, which are related to different classes including the manifestation and the subclasses of the complex event class, makes the ontology reasoning-ready for different purposes, such as configuration planning or context recognition in multi-inhabitant environments. The example scenario outlined in Section 2 is only one among a multitude of activities that are relevant to detect in the context of the E-Care@home project.

## 5. Discussion

In this paper, we are proposing an ontology, called SmartHome ontology, to formally represent different aspects of smart homes (e.g., sensing process, network configuration, objects' taxonomy, event representation, topological representation, etc.). Due to the dependencies between these aspects, the ontology could become complex and difficult to extend. We applied the ODP approach as an incremental methodology in designing ontologies in

order to deal with the complexity design issues. ODP allows us to start by first creating general and small patterns for each aspect and then links them together. In this way, regardless of its size, the ontology becomes flexible for further updates with the least possible change cost.

The majority of the ontology modules building the SmartHome ontology are relying on SSN, however with a number of modifications applied due to some representational ambiguities. These modifications are either in the form of extension of class hierarchies or updating (even removing) links between concepts. For instance, we decided to ignore the class `SSN:Observation` and only consider `SSN:Sensing` to represent an observation process. To include the essential physical and conceptual aspects of smart environments, moreover, we have extended the representation of many classes in SSN including `DUL:PhysicalObject`, `DUL:PhysicalPlace`, `DUL:InformationObject`, `DUL:Event`, `SSN:Deployment`. These extensions have made the SmartHome ontology as a generic (reusable) model to represent the evolution of a smart environment as the result of interactions between entities including smart objects, robots, humans and the other agents that one may found there.

However, reusing existing vocabularies from SSN or DUL was not a straightforward process. There are a number of classes and properties that are not strictly required in the target context (e.g., generic concepts such as `InformationObject`), however, their definition allows them to be as the basis for more specific concepts. For instance, in our model, the class `InformationObject`) is used as the super class of the two other classes, *SmartHomeFeatureOfInterest* and *State* which are used to define a situation (or more specifically, the class *SmartObjectSituation*), regardless of how it is concretely realized. A means to support the use of existing vocabularies (including both concepts and object properties) and/or patterns without the need for creating or importing such classes would ease the process to a considerable extent.

Regarding the complexity of the representation model, we have used existential quantification as the property restriction in the definition of most concepts. Although using universal quantifiers or cardinality restrictions could make the concept definitions more strict and precise, we decided to represent the patterns based on the `OWL 2 EL` profile so as to exploit its polynomial inference properties. The same justification applies for not using the disjunction operator between classes.

## 6. Conclusion & Future Work

The SmartHome ontology offers a representation model which is composed of a number of patterns representing different features of a smart environment. The representation of all the ODPs relies on SSN. However, in order to deal with the lack of precise representation suited for a smart environment, we have modified the definition of classes by either removing or updating their properties. For instance, in SSN there are alternative ways of representing the same information (e.g., observation versus sensing). SmartHome ontology is designed to provide only one possibility of representing features in order to make the extensions of the patterns, and subsequent datasets, more compatible and reduce the risk of errors in the models and data.

The modules furthermore enrich the SSN representation by extending SSN based on the general requirements of smart homes. For instance, the representation of objects, network configurations, places, events and sensing process extracted from SSN, have been extended in the form of separate ontology patterns in SmartHome ontology. Overall the ODPs proposed also forms a pattern language for smart homes, where individual ODPs can be reused without necessarily reusing the whole network of modules.

The SmartHome ontology may in the future be equipped with more patterns required to cover other aspects of a smart environment. One aspect that we are currently investigating concerns computing the number of people existing in a given smart home. One of the purposes behind modelling a smart home is to enable systems to automatically discover and then monitor the user's activities. Although the current representation model supports context inference in multi-user smart homes, reasoning about the exact number of inhabitants gives rise to ambiguities. As a next step, we will extend the ontology network to provide the required representational basis for the computational modules tracking number of inhabitants.

Table 1
Linked modules in SmartHome ontology.

| Module(1) Name | Module(2) Name | Axiom |
|---|---|---|
| SmartHome_Network | SmartHome_Place | SmartHomeDeployment ⊑ ∃ deployedOnPlatform.SmartHome |
| SmartHome_Network | SmartHome_Object | NodeStation ⊑ ∃ onPlatform.NodeHolder |
| SmartHome_Event | SmartHome_Object | SmartHomeEvent ⊑ ∃ hasParticipant.SmartHomeAgent |
| SmartHome_Event | SmartHome_TimeInterval | SmartHomeEvent ⊑ ∃ isObservableAt.SmartHomeTimeInterval |
| SmartHome_Event | SmartHome_TimeInterval | EventCondition ⊑ ∃ isObservableAt.SmartHomeTemporalDistance |
| SmartHome_Event | SmartHome_Situation | Manifestation ⊑ ∃ isEventIncludedIn.SmartObjectSituation |
| SmartHome_Event | SmartHome_Situation | EventCondition ⊑ ∃ isSettingFor.SmartObjectSituation |
| SmartHome_Object | SmartHome_Event | SmartHomeAgent ⊑ ∃ isParticipantIn.SmartHomeEvent |
| SmartHome_Object | SmartHome_Place | SmartHomeObject ⊑ ∃ hasLocation.SmartHomeSection |
| SmartHome_Object | SmartHome_Network | NodeHolder ⊑ ∃ attachedSystem.NodeStation |
| SmartHome_Place | SmartHome_Network | SmartHome ⊑ ∃ inDeployment.SmartHomeDeployment |
| SmartHome_Place | SmartHome_Object | SmartHomeSection ⊑ ∃ isLocationOf.SmartHomeObject |
| SmartHome_Sensing | SmartHome_FeatureOfInterest | SensingProcess ⊑ ∃ describes.SmartHomeFeatureOfInterest |
| SmartHome_Sensing | SmartHome_Network | SmartHomeSensor ⊑ ∃ isPartOf.SenderNodeStation |
| SmartHome_Sensing | SmartHome_TimeInterval | SmartHomeSensorOutput ⊑ ∃ isObservableAt.SmartHomeTimeInterval |
| SmartHome_FeatureOfInterest | SmartHome_Object | SmartHomeFeatureOfInterest ⊑ ∃ isAbout.SmartObject |
| SmartHome_FeatureOfInterest | SmartHome_Sensing | SmartHomeFeatureOfInterest ⊑ ∃ isDescribedBy.SensingProcess |

## References


[1] Nguyen, T. V. and Lim, W. and Nguyen, H. and Choi, D. and Lee, C.: Context Ontology Implementation for Smart Home. CoRR abs/1007.1273, (2010)

[2] Gangemi, A. and Presutti, V.: Ontology Design Patterns. Handbook on Ontologies. International Handbooks on Information Systems (2009)

[3] Blomqvist, E. and Hitzler, P. and Janowicz, K. and Krisnadhi, A. and Narock, T. and Solanki, M.: Considerations regarding Ontology Design Patterns. Journal Semantic Web, 7, 1-7 (2016)

[4] Alexander, C. and Ishikawa, S. and Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press (1977)

[5] Roger D. Eastman; Craig I. Schlenoff; Stephen B. Balakirsky; Tsai Hong Hong: A Sensor Ontology Literature Review.(NISTIR) - 7908 (2013)

[6] Tao, G. and Xiao, H.W. and Hung, K.P. and Da, Q.Z.: An Ontology-based Context Model in Intelligent Environments. Int. Conf. on Communication Network and Distributed Systems Modeling and Simulation - 270–275 (2004)

[7] Nguyen, T.A. and Raspitzu, A. and Aiello, M.: Ontology-based office activity recognition with applications for energy savings. Journal of Ambient Intelligence and Humanized Computing - 667–681 (2014)

[8] Compton, M. and Barnaghi, P. and Bermudez, L. and Garcia-Castro, R. and Corcho, Ó. and Cox, S. and Graybeal, J. and Hauswirth, M. and Henson, C. and Herzog, A. and Huang, V. and Janowicz, K. and Kelsey, W. D. and Phuoc, D. L. and Lefort, L. and Leggieri, M. and Neuhaus, H. and Nikolov, A. and Page, K. and Passant, A. and Sheth, A. and Taylor, K.: The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. Web Semantics: Science, Services and Agents on the World Wide Web. Elsevier, 17, (2012).

[9] Sezer, O. B. and Can, S. Z. and Dogdu, E.: Development of a smart home ontology and the implementation of a semantic sensor network simulator: Int. Conf. on Collaboration Technologies and Systems (CTS), (2015)

[10] Alirezaie, M. and Loutfi, A.: Reasoning for Improved Sensor Data Interpretation in a Smart Home. ARCOE-Logic Workshop Notes, pp. 1-12, (2014)

[11] Cox, Simon J.D.: Ontology for observations and sampling features, with alignments to existing models. Semantic Web Journal pp. 1-18, (2016)

[12] Janowicz, K. and Compton, M.: The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology. Workshop on Semantic Sensor Networks, SSN (2010)

[13] Seydoux, N. and Drira, K. and Hernandez, N. and Monteil, T.: IoT-O, a Core-Domain IoT Ontology to Represent Connected Devices Networks: Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW), 561–576, (2016)

[14] Battle, R. and Kolas, D.: Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL. Semantic Web Journal pp. 355–370, (2012)

[15] Cohn, A. G. and Bennett, B. and Gooday, J. and Gotts, M.: Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. GeoInformatica, 1, 275-316, (1997)